


[Web](#) [Images](#) [Groups](#) [News](#) [Froogle](#) [more »](#)

[Advanced Groups Search](#)
[Preferences](#)
 -

The "AND" operator is unnecessary -- we include all search terms by default. [\[details\]](#)

The "AND" operator is unnecessary -- we include all search terms by default. [\[details\]](#)

Groups Search result 1 for hypervisor and shared emulator OR emulation OR emulating OR emulates

NET+ARM Processors • NetSilicon's family of 32-bit RISC processors-embedded network device • www.netsilicon.com

Sponsored

[Links](#)

Real Mode • Real time multitasking for x86 embedded systems. BSP. Royalty Free • www.smxrtos.com

Grid Computing Software • distributed resource management for computer grids and clusters • www.axceleon.com

From: [324 Richard Wilmot \(rbw00@ccc.amdahl.com\)](mailto:rbw00@ccc.amdahl.com)

Search Result 1

Subject: Re: The Intel '586 and self-virtualizing

Newsgroups: [comp.arch](#)

View: [Complete Thread \(15 articles\)](#)

Date: 1992-06-19 18:52:22 PST

[Original Format](#)

In article <1992Jun18.190408.16802@ico.isc.com> dougp@ico.isc.com (Doug Pintar) write
 >It's interesting to see all the banter about "self-virtualizing" CPUs here.
 >There is absolutely *nothing* that prevents *existing* 386/486 processors
 >from running a "virtual machine" environment with a **hypervisor**. (In fact,
 >this has already been done.) The x86 privilege levels allow one to execute
 >guest OS code the same way the original VM/370 did. What *appears* to be a
 >fully-privileged execution mode to the guest OS is, in fact, a user-type mode
 >that traps all the privileged instructions. These are then interpreted and
 >emulated by the hypervisor as needed to keep the guest OS happy. The *real*
 >problem of doing this kind of thing on an x86 is the I/O architecture. On
 >the 370, a program built up a channel program and then issued an SIO to get
 >it started. The **hypervisor** merely had to translate the channel program to
 >virtualize the I/O. On the x86, you'd have to have a method of describing
 >all the virtual devices you wanted to support, so it'd have some way of
 >handling all the IN/OUT instructions to the various I/O addresses and
 >accesses to shared memory areas. A major distasteful task, ifn you ask me,
 >and one that would ultimately make an x86 **hypervisor** arbitrarily large.
 >DLP

So can I run SCO Unix under a **hypervisor**? OS/2? Windows in enhanced
 (386) mode? I was told I couldn't but would certainly be happy to
 learn how. Can my **hypervisor** maintain phantom page tables for the
 guest (Os/2, Unix, et al) operating systems? I want them to think
 they're dealing with real hardware and they will, of course, need to
 use 32 bit instructions (so NOT in V86 mode). Wont there be a problem
 when a guest operating system or one of its applications programs
 issues a RET instruction to a LOWER privelege level? This will not
 cause an interrupt so my **hypervisor** will have no opportunity to
 patch things together. There may be other problems as well.

As to **emulating** I/O, that would not seem as difficult. I run Stacker
 which does a very nice job of pretending to be a disk device
 (a driver for MS-DOS). I have less trouble with the virtual devices
 than the real ones (can't configure more than 130 MB on my wife's
 machine due to old ROM BIOS). Anyway I/O permission maps will work fine
 for masking which devices truly belong to a V86 mode guest operating sytem

✂ ✂

and which ports will cause interrupts (which we will either emulate or reflect to the guest as inoperable).

--

Dick Wilmot | I declaim that Amdahl might disclaim any of my claims.
(408) 746-6108

[Google Home](#) - [Advertising Programs](#) - [Business Solutions](#) - [About Google](#)

©2004 Google